UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/924,272	08/07/2001	Paul Metzgen	174/193	4896

36981 7590 04/22/2005

FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 AVENUE OF THE AMERICAS FL C3
NEW YORK, NY 10020-1105

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	

DATE MAILED: 04/22/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/924,272	METZGEN, PAUL	
	Examiner	**Art Unit**	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 23 November 2004.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-37 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-37 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1)☐ Notice of References Cited (PTO-892)

2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3)☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4)☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5)☐ Notice of Informal Patent Application (PTO-152)

6)☐ Other: _____.

TV HL

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 11/23/2004.

As indicated in Applicant's response, claims 1, 20, 24, 27, 34 and 37 have been amended. Claims 1-37 are pending in the office action.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.
>
> Note: 35 U.S.C. § 102(e), as revised by the AIPA and H.R. 2215, applies to all qualifying references, except when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. For such patents, the prior art date is determined under 35 U.S.C. § 102(e) as it existed prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. § 102(e)). The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that fonn the language.

3. Claims 1, 2, 4-5, 7-S, 24-26, and 34-36 are rejected under 35 U.S.C. 102(e) as being anticipated by Panchul et a1., USPN: 6226,776 (hereinafter Panchul).

As per claim 1, Panchul discloses a mcthod for generating hardware configuration data from software constructs, the method comprising: parsing high-level software programming code, wherein the code is transparent with regard to the hardware resources and hardware configuration (e.g. Fig. 2; Fig. 3-22 - Note: compiling a C prop-am instructions to provide hardware definition language and register transfer form code is equivalent to parsing and that the

C-instructions are abstract data structures remote to the layer of hardware resources, hence transparent to hardware resources and configuration); and compiling hardware configuration data directly from the high-level software programming code (e.g. Figs. 3-22; *ANSI C ... compiled into* - col. 13, lines 32-37).

As per claim 2, Panchul discloses hardware configuration data implementing blocks (e.g. HDL - col. 13, line 1 to col. 14, line 65 - Note: the organizing of hardware circuitry via HDL or Verilog specification implicitly discloses organizing the hardware configuration data in blocks - see Panchul, col. 8, lines 36-40, *always block* - Fig. 8B).

As per claim 4, Panchul discloses pipelining (e.g. Fig. 8B)

As per claim 5, Panchul discloses a FPGA (e.g. col. 12, lines 42-49; col. 13, lines 53-63).

As per claim 7, Panchul disclose runtime decisions for parallel execution (e. g. Fig. 3A-B; Fig. 14A-B; *HDL functional blocks ... independently of each other* - col. 21, line 28-46 - Note: compiling into hardware configuration, determine which HDL blocks for being synchronously executing reads on hardware blocks being generated and making independent runtime decisions while parallel executing).

As per claim 8, Panchul discloses C code (e.g. Fig. 2, 3A, 4A, 5A, 6A).

As per claim 24, Panchul discloses a method for mapping software constructs directly into hardware constructs, the method comprising

mapping software constructs directly into a block of logic operations using a software/hardware compiler (e.g. Fig. 2; *ANSI C ... compiled into* - col. 13, lines 32-37);

coupling input to the block and coupling output to that block (e.g. *Ram32xl6, Ram32xl, DivEx* - Fig. 5B1; Fig. 7B3).

As per claims 25 and 26, Panchul discloses input and output environment including wires implementing variables, pointer, expressions, and *reset* and *done* signal (e.g. Fig. 3-22; *if(reset)* - Fig. 7B3; *state=* ... , *end*- Fig. 7B3) and control flow (*controlflow* - col. 7, lines 22-43;Fig. 5A-B).

As per claims 34-36, these are programmable logic resource respective versions of claims 24-26, the programmable limitation being addressed by Panchul's programmable logic (col. 12, lines 42-49; col. 13, lines 53-63); and are rejected with the corresponding rejection in claims 24-26 as set forth therein.

4. Claim 20 is rejected under 35 U.S.C. 102(e) as being anticipated by Killian et al., USPN: 6,477,683.

As per claim 20, Killian discloses optimizing hardware generated by software-to-hardware compiler during compilation, comprising:

locating at least one expression wherein the expression is used more than once, using a single set of hardware resources to implement such expression (e.g. *pattern, replaced ... single instruction, how often, hardware estimator, set of TIE instructions* - col. 19, line 47 to col. 20, line 9; col. 27, lines 10-16- Note: detecting pattern in tree so that a set of instructions from hardware estimator can replace to improve hardware efficiency reads on using a single set of hardware resources to implement the expression if expression is used more than once); and

selecting at runtime instances that will have access to the single set of hardware resources (e.g. col. 19, line 47 to col. 20, line 9; col. 27, lines 10-16 - Note: the partitioning of hardware in blocks by a HDL mapping and optimizing replacement reads on analysis of a runtime eventuality

wherein executing instruction instances that will access a set of hardware resources as taught above).

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 3, 27, and 37 are rejeded under 35 U.S.C. 103(a) as being unpatentable over Panchul et al., USPN: 6,226,776.

As per claim 3, Panchul does not disclose data wires and a computed wire to denote the variable is valid. Panchul discloses wires being mapped to variables and associated pointers to represent a variable, expression or a function (e.g. col. 5, line 4-45; Fig. 11A-B) and a set of wires to enable a variable (LCD) with wire specialized to represent data (e.g. Fig. 20B2, *input ... A or D or WE, wire [4:0]* - Fig. 5B1), hence has suggested the creating of a wire or pointer structure representing a declared variable or function and wires for data and input control bit. Further, official notice is taken that a set of data being passed through a memory port or a circuit gate and such passage is being enabled by a valid or enable Bit was a known concept in the m4 of hardware design; and this can be shown as example in Panchul's pin WE (gate *FDCE* - Fig. 23C-2, Fig. 25C-1). Hence, since a set of wires or pointers are used for representing data or variables as suggested by Panchul's HDL and its parsing scheme, it would have been obvious for a skill in the art to represent circuitry wires according to HDL specification as suggested by Panchul above so that some bit in the set of data (or one wire among the set of wires) would be

representing the existence or validity of the variable or a non-null pointer, and that some other bit/wires represent the data for that variable according to the well-known concept above. The motivation would be because, wires are purported for transporting data through a gate device and this requires control and according to the above well-known concepts, such control, using one bit or wire to enable or indicate validity/permissibility, would allow such data lines or wires to be checked by the device or gate before data can be allowed to go beyond such gate, or to be put for storage, or read/written via memory port; and this is according the well-known concept of the enabling pin as illustrated by Panchul's above example.

As per claim 27, Panchul discloses a method of mapping software constructs into hardware constructs, comprising parsing the software constructs (e.g. col. 23, lines 63-65); mapping a software variable into a set of wires (e.g. col. 5, line 4-45; Fig. 11A-B, Fig. 20B2; *input ... A or D or WE, wire [4:0]* - Fig. 5B1). The limitation about the wires representing a variable definition/computation state and the value of the variable in this claim corresponds to that of claim 3, hence is rejected using the same rationale as set forth therein.

As per claim 37, this is the hardware and programmable logic version of claim 27, hence is rejected using the same rationale as set forth therein; the programmable limitation being addressed by Panchul's programmable logic (col. 12, lines 42-494 col. 13, lines 53-63).

7. Claims 6, 9-12, 16-19, and 28-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Panchul et al., 6,226,776, in view of Killian et al., USPN: 6,477,683.

As per claim 6, Panchul only discloses optimizing and pipelining of loops (e.g. col. 4, lines 51-63; col. 27, Table, lines 45-55; Fig. 8B); but does not teach hardware capable of speculative execution. The method of applying high-language programming optimization when

implementing compiler analysis to support pipelining and loop scheduling as endeavored by Panchul (e.g. col. 4, lines 61-67; Figs. 19, *optimized* – table, lines 45-55) was further enhanced by speculative execution of Killian. Killian, in a method to compile C-code specifications and convert HDL language into target hardware implementation analogous to Panchul, discloses pipelining as well as speculative by compiler directives and shared memory (e.g. col. 13, line 44 to col. 14, line 4). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the optimization techniques by Panchul the compiler-driven speculation as by Killian because according to Killian, these speculative executions can avert data fetching exceptions which would otherwise require unwanted correction handling resources (col. 13, line 44 to col. 14, line 4).

As per claim 9, Panchul discloses a method for generating hardware exploiting parallelism by making decisions at runtime, the method comprising:

generating a control flow in the hardware configured in blocks (e.g. control flow, HDL, Verilog - col. 7, lines 22-43; Fig. 5A-B; col. 8, lines 36-40; *always block* - Fig. 8B);

making determination to exploit parallel execution based on HDL compilation and configuration, i.e. based on control flow, event or machine state - *event* and *state* read on dynamic behavior or runtime analysis- associated with HDL analysis (e.g. Fig. 3A-B, Fig. 12A-B, 14A-B).

But Panchul does not disclose that the control flow indicates a status for a block, status indicative for a capability for speculation. The concept as to base on an entry tag value, a control bit, a variable value, or predicate check in order to establish whether a chunk of instructions can be executed speculatively via compiler analysis was a known concept as has been evidenced

from Killian's optimization using speculative execution from above. The limitation as to provide speculation to the implementation from compiling HDL into hardware target architecture set has been addressed above using Killian; hence this speculation based on a block status while analyzing control flow as mentioned above by Panchul would have been obvious in light of the known concept and of Killian's teachings for the same reasons as set forth in claim 6 above.

As per claims 10 and 11, the partitioning of HDL constructs into blocks of instructions as evidenced by Panchul (col. 8, lines 36-404 always block - Fig. 8B) was a known concept in the HDL programming language (e.g. Verilog, VHDL, Handel-c) and in view of Panchul's teaching of determining to execute simultaneous instructions based on event or machine state analysis or to always execute a block (Fig. 3A-B; Fig. 12A-B; 14A-B; Fig. 8A) ms put forth in claim 9, the rationale to use an indication at a given point into a block of instructions in order to determine whether to execute the block via speculation has been addressed in claim 9; hence the rationale as to combine Panchul' s block with well-known concepts in association with the speculative execution as taught by Killian is herein applied to address why it would be obvious to decide to execute the block based on a tag or predicate statement evaluation.

As per claim 12, the rationale for rejection has been addressed in claim 9.

As per claim 16, Panchul teaches always blocks, hence has implicitly disclosed them to be non-mutable operations (e.g. Fig. l2B).

As per claim 17, Killian discloses replacing multiple patterns with a single instructions (col. 1 9, lines 47-65) and replacing a maximally sized match in the functions tree with one equivalent instruction (e.g. col. 27, lines 10-16). Hence, in the line as to enhancing further of

Panchul's teaching of a tree traversal and pipeline techniques, it would have been obvious for one of ordinary skill in the m4 at the time the invention was made to provide on top of the variable mapping and function creation as suggested by Panchul (Fig. 3-22) the replacement techniques (or mutable instruction representing a HDL state bcing replaced) by Killian because of the same reasons as to optimize resources being recognized as well-known at the time the invention was made in the art of compiler where a excess or potential redundant resource usage is averted by a replacement policy or instruction just as taught by Killian.

As per claim 18, Panchul discloses a control flow (e.g. Fig. 5A-B).

As per claim 19, Panchul discloses implementing software in hardware (e.g. control flow, HDL, Verilog - col. 7, lines 22-43,. Fig. 5A-B,' col. 8, lines 36-404 always block - Fig. 8B).

As per claim 28, this is a programmable logic resource version of claim 9, the programmable limitation being addressed by Panchul's programmable logic (col. 12, lines 42-49; col. 13, lines 53-63); and is rejected with the corresponding rejection in claim 9 as set forth to address the control flow status for an operation and decision partially based on the control flow limitations.

As per claims 29-33, these claims correspond to claims 10-12, 18, and 19 respectively; hence are rejected using the corresponding rejection as set forth therein respectively.

8. Claims 13-15, and 21-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Panchul et al., USPN: 6,226,776, in view of Killian et al., USPN: 6,477,683; and further in view of Ashar et al., USPN: 6,745,160 (hereinafter Ashar).

As per claims 13-14, Panchul does not explicitly teach deciding to execute the block speculatively and in parallel; or no data dependency between speculative and parallel execution of blocks. But Panchul discloses a tree of function/nodes (Figs. 11) and determining if an external event or a state machine is required during analysis of called function/node derived from HDL and if not then simultaneous execution of C-type functions can be effected (e.g. col. 20, line 22 to col. 21 , line 17); and non-dependency of C-type functions being simultaneously executed (e.g. col. 21, lines 28-46); hence has taught no data dependency of instructions blocks (or functions from a tree node) being paralleled.

The motivation as to use speculation in optimization of runtime execution of functions has been addressed using Killian. As for the decision to execute the block in parallel or speculatively, Ashar, in a method using netlist and VHDL to transform loop execution into a optimized and validated sequence or scheduling using speculation and parallelism as mentioned by Panchul and Killian, further discloses pipelining loops and speculative execution being determined upon analysis of control or state graph, loop invariants, and state and boundaries verification (e.g. col. 26, lines 17-29; col. 27, line 45 to col. 28, line 6). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the combination Panchul/Killian the verification ms taught by Ashar so that decision can be made to use either speculation or pipelining or loop unrolling because Ashar's techniques would provide timely validating of data being fetched for execution thus enhance optimizing success without sacrificing resources for data checking and dependencies when it is too late in the execution stage of complex loop execution (see Ashar Background).

As per claim 15, Panchul discloses resource sharing with re-used blocks (col. 5, line 56 to col. 6, line 4) and Killian sharing of a memory portion (col. 13, line 44 to col. 14, line 4). In view of the teachings by Ashar and the non-dependency teaching by Panchul as mentioned above, it would have been obvious for one of ordinary skill in the a14 at the time the invention was made to provide on top of such non-dependency when running in parallel and/or via speculation the step of sharing a variable in memory or a block as suggested by Panchul and enhanced by Killian to the method of optimization provided by Panchul/Killian and Ashar; because of the same reasons as to optimize resources so well known at the time the invention was made in the art of loop pipelining and parallel execution of instructions in the art of compiler optimization techniques.

As per claim 21, in reference to claim 15, Panchul discloses using a single set of hardware resources to implement a software program entities represented by a node in a control flow, i.e. a set of hardware resources while converting a HDL function or block; into hardware implementation (e.g. Fig. 3-22).

As per claims 22 and 23, Panchul/Killian and Ashar implicitly discloses the optimization process, such as pipelining, speculation execution etc. to take place late in the phases of HDL conversion without intervention of the user (re claim 1 5).

Response to Arguments

9. Applicant's arguments filed 11/23/04 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

Rejection under 35 USC § 102 (e):

(A) As per claims 1, 24, 34, Applicant has submitted that Panchul does not directly map high language constructs into hardware configuration data but needs a intermediate definition language which in turn has to be compiled in configuration data (Appl. Rmrks, pg. 17, 2nd para). The claim merely states converting directly into hardware configuration data; and as interpreted this data does not enforce hardware per se nor does it preclude a HDL being compiled into some more concrete hardware entities. The limitation 'configuration data' encompasses so much into it so that merely asserting that it represents or has to be equated to a particular form of hardware entity would not be convincing. In the rejection, portions are cited for showing that C language constructs are mapped into some data of the HDL; and this latter language being thus created is one of many interpretations that one skill in the art would adopt when construing the afore-mentioned 'hardware configuration data'. Hence, the Applicant's argument is deemed not convincing.

(B) As per claim 20, Applicant has submitted that Examiner asserts 'inherently yields runtime instances' (Appl. Rmrks, pg. 19, 2nd para; pg. 20, middle); and this assertion is unfounded and that Killian only teaches compilation time. The rejection now is pointing to the fact that the meaning of 'selecting runtime instances ...' has been viewed as analysis by the compiler for a runtime eventuality wherein patterns that are repeated can be replaced. In other words, as construed from the claim language which starts from a '... compiler during compilation' and ends with 'selecting at runtime instances ...', for one skill in the art the understanding is that the *selecting* step is based on analysis performed by the compiler and there is use of some compilation directives to replace patterns that would be considered repeated at runtime; and to perform such replacement. And this is what Killian's block or pattern

replacement is done during the tree flow analysis. The claim does not establish a clear/explicit context that enforces a unique situation in which the step of selecting is necessarily done by a code being executed (i.e. runtime), wherein it is this very code that selects. As recited, no code is being executed and only a method for optimizing is claimed. Such method comprises 'locating …', 'using …' and 'selecting at runtime instances'. This method clearly does not enforce that 'runtime' is runtime of one unequivocal process. Hence, runtime can be either runtime of a compiler optimization program or that of the code created from such compiler; whereas 'method of optimizing' bears heavily on the connotation that it is actually the compiler doing the optimization and it is such compiler process that envisages some runtime instances in which data can be optimized. And as such Killian cited portions have met the limitation; and this is put forward in the rejection. As a result, the argument about inherency does become moot.

Rejection under 35 USC § 103 (a):

(C) As per claims 27 and 37, Applicant has submitted that an official notice along with Examiner's combination does not fulfill the limitation about variables as a set of wires (Appl. Rmrks, pg. 21, 2nd para); and there is no motivation to combine (Appl. Rmrks, pg. 22, 2nd para) for hindsight has been improperly applied (Appl. Rmrks, pg. 23, top). This rejection herein is pointing to where some hardware pin derived from Panchul's HDL and being imparted to logic gate representation can be used to dictate validity of a variable or variables representing a set of wires. In terms of computer executing a logic as Panchui describes via set of wires of a gate, there is no clear separation between a set of wires and a set of variables because the computer will operate of digital data representing machine representation of those variables implementing the HDL definition of wires; and the motivation to combine what Panchul already has (see pin

WE in cited portions) with the concept of turning on a wires (a variable) to validate a logic input/output (also variables) has been set forth as obvious. For one skill in the art, if the reference contains sufficient teaching as to make some reasonable readjusting of the reference so that what is not explicitly disclosed therein can be modified for a good purpose without adverse effect, then the act of combining and modifying such as has been set forth in the rejection is not considered hindsight so long as the facts are found in the very prior art at hand. The argument is considered not convincing further because the Applicant has yet to convey specifics as to why the combined teachings as set forth in the rejection would not lead to expected good results.

(D) As per claims 9 and 28, Applicant has submitted that Panchui does not teach a status indicative of a block's capability for speculation; and that Panchui fails to teach a runtime decisions making (Appl. Rmrks, pg. 25-26). The limitation as to make runtime decisions has been addressed in section B above. And the limitation as to enable speculation, Panchul discloses a execution using all the capabilities of high-level language programming to provide optimization, such teaching being pointed in the rejection; and this is leading to a rationale as to render obvious the speculative execution as required by the claims. Specifically, claim 9 is using the rationale from claim 6 which addresses why optimization in complex pipelining would be enhanced via speculative execution; and Applicant has yet to put forward specifics as to why the combined teachings as set forth in the rejection of claim 6 would not lead to expected good results; otherwise the argument about a prima facie case of obviousness rejection not being established would not stand. Further, the argument that the combination from Panchul and Killian is merely hindsight is hence referred back to section C above.

Conclusion

10. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
April 9, 2005

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100